CHALMERS
UNIVERSITY OF TECHNOLOGY

Comparison of Exact and Approximate methods for the Vehicle Routing Problem with Time Windows

Comparison of Exact and Approximate methods for the Vehicle Routing Problem with Time Windows

Elinor Jernheden[1] and Carl Lindström[1] and Rickard Persson[1] and Max Wedenmark[1]
Endre Erős[2] and Sabino Francesco Roselli[2] and Knut Åkesson[2]

***Abstract*—This paper presents a comparison of two approaches for solving the vehicle routing problem with time windows (VRPTW). Scheduling of vehicles for pickup and delivery is a common problem in logistics and may be expressed as VRPTW, for which both exact and approximate techniques are available. It is therefore interesting to compare such techniques to evaluate their performance and figure what is the best option based on the instance features and size. In this work, we compared Mixed Integer Linear Programming (MILP) with Set-Based Particle Swarm optimization (S-PSO). Both algorithms are tested on the full 56 instances of the Solomon dataset. The results show that the two algorithms perform similarly for lower number of customers while there are significant differences for the cases with higher number of customers. For higher number of customers MILP consistently performs as good as or better than S-PSO for the clustered data, both with short and long scheduling horizons, while the S-PSO outperforms MILP in most cases with random and mixed random clustered data with long scheduling horizons. Furthermore when the algorithms perform the same with regards to the main objective (number of vehicles), MILP generally achieves a better result in the second objective (distance traveled).**

I. Introduction

This paper presents and compares two approaches for solving the vehicle routing problem with time windows (VRPTW). The VRPTW is a NP-hard combinatorial optimization problem [1] that is widely employed to model transportation and logistic systems [2]. The problem formulation aims at selecting the best available routes for a set of vehicles such that all customers are visited and serviced. A broad range of routing, pickup and delivery problems can be expressed as VRPTW and an efficient approach for the VRPTW may be a valuable asset. There exist different algorithms to solve the VRPTW [3]. It is therefore interesting to compare these techniques to asses their effectiveness in providing feasible optimal and sub-optimal solutions to a particular class of instances.

The VRPTW is a linear minimization problem where the search space may be defined as a directed complete graph since it naturally represents the spacial distribution of the customers and can use nodes and edges to specify the different properties of customers and routes. The VRPTW is then solved by choosing the most suitable subset of arcs. In order to solve the problem optimally it is necessary to handle the inherit property of choosing between using or not using each arc. Being the problem intrinsically linear, a suitable option is to model it as a mixed integer linear programming (MILP) problem, defining a set of binary variables, which can be used to model the decision making. Many different MILP models have been developed for different versions of the vehicle routing problem (VRP) . Common approaches are models based on three indexes (from **a** to **b** using vehicle **v** representation) as presented in [4] and models based on two indexes (from **a** to **b** representation), as recently presented in [5]. Also, when it comes to the objective function, the cost criterion is usually to minimize either the number of vehicles or the traveled distance, or a combination of both, implemented by using a weighted sum.

On the other hand, due to the computational complexity of the problem, time increases strongly with higher number of customers and research hence started focusing on approximate methods to achieve sub-optimal solutions in shorter time compared to the exact methods, such as the tabu search method proposed by Gmira *et al.* [6], the ACO based approach from Li *et al.* [7], the local search algorithm by Arnold [8] and the genetic algorithm approach by Abbassi *et al.* [9].

Another approximate method which prooved effective is Set-based Particle Swarm Optimization (S-PSO), proposed in [10], based on the continuous stochastic optimization algorithm Particle Swarm Optimization (PSO). PSO, introduced in [11], has become an widely used algorithm for solving continuous problems [12], [13]. However, as the VRPTW is defined in the discrete search space, modifications are needed for the PSO to be applicable.

As for the MILP approach, the model formulation presented in [5] provides a compact representation that leads to improvement in the performance compared to other MILP formulations; Also, the S-PSO performance proves competitive among the approximate methods. Being these methods among the state-of-the-art approaches in their respective categories, it is of interest to compare them to determine their strengths and weaknesses. They are benchmarked using all 56 instances of the Solomon dataset [14]. Each instance is defined by a depot, located in the origin of the reference system and a set of customers, each identified by a number (ID), a location (in bi-dimensional Cartesian coordinates) and a time-window. The dataset is available on http://w.cba.neu.edu/~msolomon/home.htm. The contributions in this paper are: (i) assess what are the options available in the literature to solve the VRPTW using both approximated and exact methods; (ii) benchmark such methods over the full set of Solomon instances; (iii) provide information about strengths

We gratefully acknowledge financial support from Chalmers AI Research Centre (CHAIR) and AB Volvo (Project ViMCoR). Department Electrical Engineering, Chalmers University of Technology, Göteborg, Sweden [1]{jelinor, carlinds, ricperss, maxwed}@student.chalmers.se [2]{endree, rsabino, knut}@chalmers.se

and weaknesses of the methods based on the instances size and features. The paper is organized as follows: Section II presents the VRPTW by introducing both an informal description as well as a formal mathematical definition of the problem. Section III provides the MILP formulation used for the VRPTW in this paper. Section IV describes the S-PSO algorithm proposed in [10]. Section V presents the numerical results for both MILP and S-PSO evaluated on Solomon's benchmarks. In section VI conclusions drawn from the results are presented.

II. The Vehicle Routing Problem with Time Windows

VRPTW, as defined in [10], is a combinatorial optimization problem which is NP-hard. The problem consists of n number of customers that need to be served by a fleet of vehicles which all start from a common depot. Each customer is denoted by an index i ($i = 1, \ldots, n$) and is also associated with a quantity q_i of goods which needs to be delivered by a vehicle for the customer to be considered served. The vehicles have an upper limit for the quantity of goods they can carry which is denoted as the capacity Q. If a vehicle exceeds the capacity Q during its route another vehicle thus has to be dispatched to serve the remaining demand of the customers. Each customer may also only be visited by exactly one vehicle i.e. multiple vehicles may not serve the same customer. Apart from the capacity constraints the VRPTW also includes time windows and each customer is further associated with a service time s_i. A vehicle consequently has to stay at a customer for a time of at least s_i in order to complete the service. The time windows are denoted $[e_i, l_i]$ and the constraint of the time windows in the VRPTW is that the service of customer i has to start within its designated time window. A vehicle then has to wait a certain amount of time, denoted as the waiting time w_i, if a vehicle arrives at a customer earlier than e_i. Similarly to when a vehicle exceeds its capacity Q if a vehicle can not arrive at a customer i before l_i then this vehicle can not serve customer i and another vehicle has to be dispatched. Even though the depot is unique in the sense that it defines the starting point of all vehicles and is also not associated with any requirements its properties is still denoted using the same notation as for the customers. The depot has a service time $s_0 = 0$, a demand $q_0 = 0$ and time window $[e_0, l_0]$ where e_0 is the earliest start time and l_0 is the latest return time of all the vehicles.

The nodes and arcs of the graph representing the problem can be used to define properties such as demands q_i, service times s_i, time windows $[e_i, l_i]$ and travel times d_{ij}. The graph can be defined as $G = (V, A)$ where $V = (c_0, c_1, \ldots, c_n)$ is the set of nodes i.e. the depot and all the customers and $A = \{\langle c_i, c_j \rangle | c_i, c_j \in V, c_i \neq c_j\}$ is the set of arcs i.e. the different paths between the nodes.

The formal mathematical definition of the VRPTW can be stated similarly as in [10] where the following variables are defined: x_{ij}^k: Binary variable that becomes 1 if vehicle k travels directly from i to j, 0 otherwise; y_i^k Binary variables that becomes 1 if customer i is served by vehicle k, 0 otherwise; $t_i \in \mathbb{R}\ i = 0, \ldots, n$ and $w_i = \max\{e_i - t_i, 0\}\ i = 0, \ldots, n$ are the arrival time and waiting time at customer i respectively and e_i is the start of the time window for customer i. The optimization problem of the VRPTW is to minimize the two objectives.

$$\min v \qquad and \qquad \min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{v} d_{ij} \times x_{ij}^k \tag{1}$$

such that

$$\sum_{i=0}^{n} x_{ij}^k = y_j^k \quad \forall k = 1, \ldots, v, \quad i = 1, \ldots, n \tag{2}$$

$$\sum_{j=0}^{n} x_{ij}^k = y_i^k \quad \forall k = 1, \ldots, v, \quad i = 1, \ldots, n \tag{3}$$

$$\sum_{i=0}^{n} y_i^k \times q_i \leq Q \quad k = 1, \ldots, v \tag{4}$$

$$\sum_{k=1}^{v} y_i^k = 1 \quad i = 1, \ldots, n \tag{5}$$

$$\sum_{k=1}^{v} y_0^k = v \tag{6}$$

$$t_i + w_i + s_i + d_{ij} = t_j \quad i, j = 0, 1, \ldots, n, \quad i \neq j \tag{7}$$

$$e_j \leq t_j \leq l_j \quad j = 0, 1, \ldots, n. \tag{8}$$

The constraints in equations (2) and (3) formulate the fundamental property that a customer or depot is served by a vehicle if a vehicle travels from or to the corresponding customer or depot. The constraint in equation (4) describes that a vehicle can not carry a quantity of goods that exceeds the vehicle capacity Q. The constraint in equation (5) denotes that each customer may only be served by exactly one vehicle. Equation (6) formulates the constraint that all the vehicles originate from the depot. Equations (7)-(8) formally define the time window constraint which also includes a definition of the arrival time t_j at customer j. Furthermore d_{ij} is the travel time between customers i and j or the travel time between a customer and the depot if either i or j is zero. However the travel time between i and j is represented by the euclidean distance between i and j and the travel time is hence interchangeable with the travel distance. The optimization objective of the VRPTW is twofold. The primary objective is to minimize the number of vehicle routes (denoted as v) and with this number of routes the secondary objective is to minimize the total distance traveled by all the vehicles. A solution of the VRPTW is a number of routes where the demand of all customers is fulfilled while at the same time not violating any of the constraints stated in equations (2)-(8).

III. Mixed Integer Linear Programming

The VRPTW is a minimization problem under a set of linear constraints and it includes logical decision making: it is therefore possible to write the problem as a MILP program, using two-index variables, as in [5].

The model therefore has three sets of variables:

$$\begin{aligned} r_{ij} &= \{r_{ij} \in \mathbb{Z} \wedge r_{ij} \geq 0 \wedge r_{ij} \leq 1\} \\ T_i &= \{T_i \geq 0 \wedge T_i \leq \text{maxTime}_i\} \\ qv_i &= \{qv_i \geq 0 \wedge qv_i \leq \text{maxLoad}\} \end{aligned} \tag{9}$$

where r_{ij} is a set of binary variables representing whether the arc from node i to node j is used. T_i represents the arrival

time of a vehicle to node i and qv_i is the load carried by the vehicle on arrival to node i. It is assumed that all vehicles start at node S and finish at node E, servicing all nodes C on their way, $N = (S \cup C \cup E)$. In the standard VRPTW the data is identical for S and E to simulate that it is the same node. The full MILP model of the VRPTW is

$$\min \sum_{i \in S, j \in N} r_{ij} + \sum_{i \in N, j \in N} \frac{d_{ij} r_{ij}}{\text{maxRoute}} \text{ s.t.} \quad (10)$$

$$\sum_{i \in C, j \in N} r_{ij} = 1, \quad (11)$$

$$\sum_{i \in N, j \in C} r_{ij} = 1, \quad (12)$$

$$p_{ii} = 0, \ \forall i \in N \quad (13)$$

$$r_{ij} = 0, \ \forall i \in N, \ j \in S \quad (14)$$

$$r_{ij} = 0, \ \forall i \in E, \ j \in N \quad (15)$$

$$T_i + s_i + d_{ij} + M_1(r_{ij} - 1) \leq T_j, \ \forall i, j \in N \quad (16)$$

$$T_i \geq 0, \ \forall i \in N \quad (17)$$

$$T_i \leq l_i, \ \forall i \in N \quad (18)$$

$$qv_i \geq 0, \ \forall i \in N \quad (19)$$

$$qv_i \leq Q, \ \forall i \in N \quad (20)$$

$$qv_i + q_i + M_2(r_{ij} - 1) \leq qv_j, \ \forall i, j \in N \quad (21)$$

$$\sum_{i \in S, i \in N} r_{ij} \leq \text{maxVehicles}. \quad (22)$$

The objective function is given by (10). The minimization of the number of vehicles precedes the minimization of the traveled distance. Therefore the traveled distance is normalized using maxRoute to ensure that it is restricted to a value between zero and one. Constraints (11)-(13) handles how vehicles flow through the graph. Constraints (14)-(15) handles the depot, which is both the start and the end of all routes. Constraints (16)-(18) is the time window constraints and (19)-(21) handles the load constraints. The amount of available vehicles is limited by constraint (22). The variable M_1 is chosen to be $\max\{b_i + d_{i,j} - a_j\}$, $i, j \in N$ as done in [15]. The variable M_2 is chosen to be the maximum capacity Q of the vehicles. The constant maxRoute is chosen as the the sum of the distance traveled if every node C were to be serviced by its own vehicle.

IV. Set-Based Particle Swarm Optimization

The S-PSO follows the same general structure as the original PSO, but instead uses operators and procedures defined on crisp sets and possibilities, using a set-based representation scheme. The set-based representation scheme is formed by mapping the original formulation from Section II, defined by a finite set of candidate solutions, a set of constraints and an objective function, to a problem of choosing a subset of elements from the universal set that optimizes the objective function. For the VRPTW, the universal set is the set of directed arcs, and the problem will be mapped to choosing a crisp set of arcs that satisfies all the constraints and optimizes the number of vehicles needed as well as the total distance traveled. Each customers location in the VRPTW will be represented by a node on the directed complete graph.

A. Particle Representation

As described in [10], a particle's position is represented by

$$X_i = \left[X_i^0, X_i^1, ..., X_i^n\right] \quad (23)$$

$$X_i^d = \left[\langle nb_{i,1}, d\rangle, \langle d, nb_{i,2}\rangle\right], \quad nb_{i,1}, nb_{i,2} \in \{0, 1, ..., d-1, d+1, ..., n-1, n\}, \quad nb_{i,1} \neq nb_{i,2} \quad (24)$$

where X_i is a set of arcs denoting a feasible solution for particle i, and n is the number of nodes in the complete graph. Each dimension of the particle, X_i^d, is constructed by two arcs that is going to and from node d. Node $nb_{i,1}$ is denoting the node previous to d while $nb_{i,2}$ is the node subsequent of node d. The velocity of a particle is represented by

$$V_i = \left[V_i^0, V_i^1, ..., V_i^n\right], \quad (25)$$

$$V_i^d = \left\{\langle u_i, v_i\rangle / p(u_i, v_i) | \langle u_i, v_i\rangle \in A^d\right\}, \quad (26)$$

where A^d is the set of adjacent arcs to node d and $p(u_i, v_i) \in [0, 1]$ is the corresponding probability for arc $\langle u_i, v_i\rangle$ to be selected in the position updating step. Any arc $\langle u_i, v_i\rangle$ with corresponding probability $p(u_i, v_i) = 0$ will be left out. At the start of the S-PSO, the particles positions are initialized by a nearest neighbour heuristic (NNH) with a probability ϕ, and randomly otherwise by sampling the order of the nodes from a uniform distribution. The NNH is employed to select the "closest" customer where closeness is defined as

$$\text{heuristic}_{km} = t_{km} + w_m + l_m - t_m \quad (27)$$

where t_{km}, w_m, l_m and t_m is defined as in section II. Thus each element in the matrix heuristic_{km} is a closeness measure between nodes k and m. The closeness measure is not only given by the physical distance between the nodes but also contains information describing the time window aspects as can be seen in equation (27). The solution obtained from the NNH is then the route for which the closest node is selected in each step, starting at the depot, where the closeness is given by the corresponding element in the matrix heuristic_{km}. The velocities for all particles are randomly initialized where both the arcs and probabilities are sampled from a uniform distribution. After initialization, each particle's position is converted into a feasible solution with the use of a constraint-based decoder. The decoder starts from the depot, and for each arc in the particle's position, check if the arc satisfies the constraints. If all constraints are satisfied, the arc is preserved, otherwise a new arc is inserted back to the depot and thus creating a new vehicle route.

B. Particle Update

The S-PSO in [10] employs a velocity update used in the comprehensive learning PSO (CLPSO), proposed by [16]. The results in [16] showed that the CLPSO has very good performance when solving multimodal optimization problems, and retains the diversity in the swarm to prevent premature convergence. The velocity update is thus defined as

$$V_i^d = \omega \times V_i^d + c \times \text{rand}^d \times \left(p\text{Best}_{f_i(d)}^d - X_i^d\right) \quad (28)$$

where c is an acceleration constant, ω is the inertia weight used to balance a particle's exploration versus exploitation, d represents the current dimension and $f_i(d)$ defines which particle's pBest the particle i should learn from for dimension d. The selection of $f_i(d)$ is done probabilistically, using a probability Pc. A random number $r \in [0, 1]$, sampled from a uniform distribution, is generated for each dimension of a particle i, if r is bigger than Pc, the particle will learn from itself for that dimension, i.e $f_i(d) = i$. Otherwise, the particle will choose another particle to learn from for that dimension using a tournament selection, as described in [16]. Further, the operations in equation (28) are defined on set and possibilities where the coefficient$\times$velocity and velocity+velocity operators are defined such that they change the corresponding probability $p(u_i, v_i)$. The position$-$position operator is defined as removing the elements in the first set that also belongs to the second set, and the coefficient$\times$(position$-$position) operator is defined such that it converts the resulting crisp set of the set subtraction into a set with possibilities.

The position is updated for each particle i, and for each dimension d as

$$X_i^d = X_i^d + V_i^d, \tag{29}$$

where the + operator has been redefined. In the position update step, the velocity is first converted into a crisp set of arcs as

$$\begin{aligned} &\mathrm{Cut}(V_i^d) = \\ &\left\{ \langle u_i, v_i \rangle \mid \langle u_i, v_i \rangle / p(u_i, v_i) \in V_i^d \text{ and } p(u_i, v_i) \geq \mathrm{rand} \right\} \end{aligned} \tag{30}$$

where rand is a uniformly sampled number between 0 and 1. Thus, an arc with higher probability is more likely to be kept in the set after the cut. The new position is then built one arc at a time, starting from the depot, and taking the constraints of capacity and time windows into account. The next arc is then selected adjacent to the current node, and new customers are selected iteratively. If possible, the next customer will be selected from the set of crisp arcs converted from the velocity, $\mathrm{Cut}(V_i^d)$. If that set is empty, the next customer will instead be selected from any arc in X_i that satisfies all constraints, and if that set is empty as well the next customer will be selected from any feasible arc in the universal set. Should all sets be empty, then a new vehicle route is inserted and the procedure starts from the depot again. Thus, all updated positions are guaranteed to be feasible. The selection in these steps could be done either using a heuristic approach, as described in equation (27), or by random selection. In addition to the position update, an additional local search is also used. For each particle, the route with the fewest customers is chosen. Each customer in the route is then inserted into the other routes, on the condition that the start and end time does not change for any customer in the route and that all constraints of the VRPTW are still fulfilled. If all customers in the shortest route can be inserted into other routes, the shortest route is deleted and the particle's position is updated. If any insertion violates the constraints or changes the start and end times, the particle's position stays the same.

C. Particle Evaluation

The VRPTW described in section II has two objectives, firstly to minimize NV (number of vehicles) and secondly to minimize TD (traveled distance). As the NV is an integer number, any two solutions with different NV must satisfy that $|\mathrm{NV}(X_i) - \mathrm{NV}(X_j)| \geq 1$. Thus if the TD is normalized within the boundaries 0 and 1, it is guaranteed that the NV will be minimized in first hand and TD secondly. The objective function used is defined as

$$\mathrm{cost}(X_i) = \mathrm{NV}(X_i) + \arctan(\mathrm{TD}(X_i))/(\pi/2). \tag{31}$$

The cost is then used to update each particle's pBest whenever a new lower cost is found. The objective function for the SPSO is designed similar to the one for MILP, with the hierarchical objective of primarily minimizing the number of vehicles used and the secondarily objective of minimizing the total distance traveled by the vehicles. Such choice is made for two reasons: it is the convention adopted when comparing different methods over benchmark instances; from an industrial point of view, it is more interesting to find solutions with fewer vehicles, since the purchase of assets can impact on the costs more than the energy consumption when travelling. Also, due to different constraints in the implementation of the algorithms, they are designed with different normalization methods for the traveled distance, but are in both cases normalized within 0 and 1.

V. Evaluation

This section presents the results from running the S-PSO and MILP algorithms on Solomon's VRPTW benchmark problems, [14]. The problems include randomly generated geographical data, marked as *R*, clustered data, marked as *C* and mixed clustered and random data, marked as *RC*. The problem sets also include short scheduling horizons, *R1*, *C1*, *RC1*, and long scheduling horizons, *R2*, *C2*, *RC2*.

A. Parameters and Settings

The S-PSO and MILP were both run on Solomon's 25, 50 and 100 customer instances. It was found during testing that limiting the solving time for the MILP model was needed to avoid extreme solving times. The behaviour of the solving process was found to be very different among the different instances. Some instances could be solved to optimality in minutes whereas many stagnated for hours. This problem is assumed to be the consequence of the solver getting stuck in a local optima and favors the addition of a solver time limit. The solver was therefore limited to a maximum solving time of 10 minutes (600 seconds) for each instance. During the evaluation of the S-PSO on Solomon's instances the parameters were set as follows. The S-PSO was run with 20 particles, and a termination criterion of 1000 generations without any improvement of the best particle. The acceleration constant c was set to 2, and the probability for NNH initialization was set to $\phi = 0.3$. The parameters for the CLPSO velocity update was set according to [16], where $f_i(d)$ is said to be updated after seven iterations without improvement. The inertia weight w was initially set to $w = 0.9$ and was linearly decreased to a minimum value of $w = 0.4$ over the run, where the step size in this linear decrement was $\Delta w = 0.0005$. The S-PSO was run for 10 evaluations of each instance, using the random selection operator. The S-PSO is coded in Python. The MILP model was solved in Gurobi version 8.1 using the Python API. Both algorithms were run on a machine running Ubuntu-18.04.3 LTS, with an Intel Core i7 6700K 4.0 GHZ and 32GB RAM.

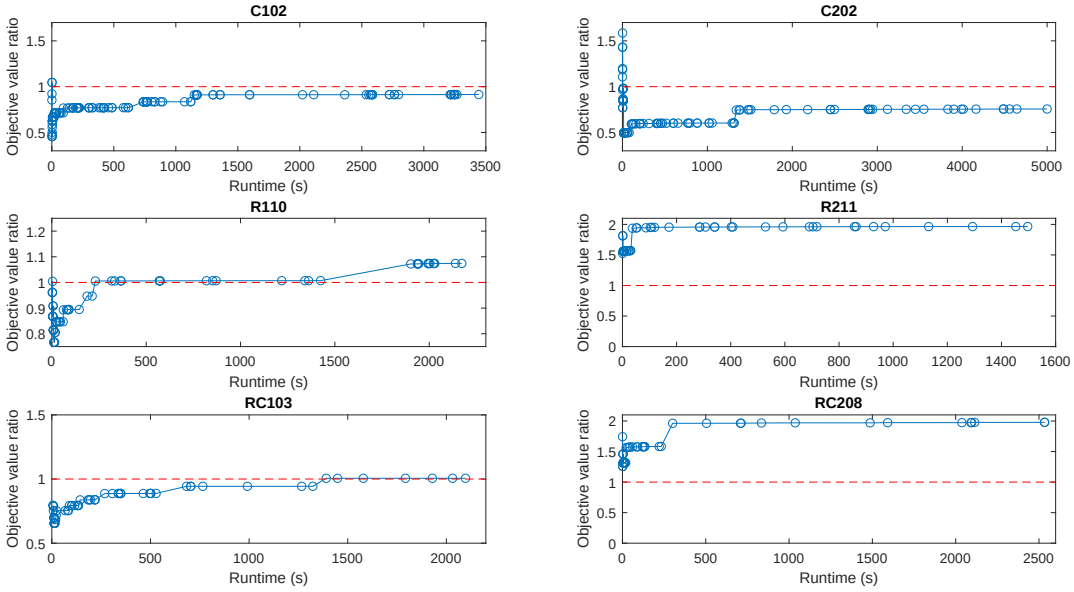

Fig. 1: Plots of the ratio of the objective values for S-PSO and MILP for six different instances with 100 customers. The ratio is given by the objective value of MILP divided by the objective value of S-PSO i.e. a ratio larger than one indicates that S-PSO found a better solution compared to MILP. The circles indicate that one of the algorithms found a better solution and the dashed red line illustrates the threshold for which one of the algorithms is performing better.

TABLE I: Comparison of running time (in minutes) of the S-PSO and MILP models for the Solomon instances. Instances are grouped in classes and the time shown is the average. Termination criterion is a time limit of 10 minutes for MILP and 1000 iterations without improvement for S-PSO. For MILP, the number of solved instances for each class is reported.

Customers	Instance	S-PSO	MILP	
		Runtime	Runtime	Optimum
25	c10	0.68	0.04	5/9
	c20	0.65	0.03	7/8
	r10	0.88	2.99	5/12
	r20	0.93	0.01	1/11
	rc10	0.76	0.13	4/8
	rc20	0.82	0.03	1/8
50	c10	4.76	0.02	4/9
	c20	4.56	0.12	2/8
	r10	4.57	0.15	2/12
	r20	5.19	1.42	1/11
	rc10	4.76	1.84	1/8
	rc20	5.11	-	0/8
100	c10	32.49	0.05	3/9
	c20	29.29	0.27	2/8
	r10	23.79	0.01	1/12
	r20	34.14	-	0/11
	rc10	25.18	-	0/8
	rc20	20.02	-	0/8

B. Comparison

The results from running Solomon's instances are summarized in Table I[1]. The S-PSO and MILP algorithms perform very similar in the 25 customer instances, especially on the problems with clustered data, *C101-C109* and *C201-C208*. Both algorithms reach the same NV for these instances, but the MILP manages to obtain a lower TD in most cases. The S-PSO is consistent in reaching the optimal NV, but varies in TD. The results are similar for the problems with random data, *R101-R112* and *R201-R211*, but the S-PSO is not as consistent in terms of NV and thus obtains a slightly higher mean than the best solution found. The MILP performs worse in two instances, *R205* and *R210*, where the S-PSO manages to find a better solution on average. For most of the instances when S-PSO and MILP obtains the same NV, the MILP still performs better on optimizing the second objective, TD. For the problems with mixed clustered and random data, *RC101-RC108* and *RC201-RC208*, MILP performs worse in three instances, *RC206*, *RC207* and *RC208*, where S-PSO manages to obtain lower NV on average. However in most instances where the NV is the same for both algorithms, MILP obtains a better TD.

The algorithms perform similar on the clustered data in the 50 customer instances as well, with MILP performing slightly better on minimizing the second objective, TD. The MILP performs better compared to the S-PSO for the random data with short horizons, *R101-R112*, where it manages to find a lower NV in six cases and a lower TD for all cases where the two algorithms reach the same NV. The S-PSO manages to perform slightly better compared to the MILP in the cases with random and mixed random clustered data with longer horizons, *R201-R211* and *RC201-RC208*, where it finds a lower NV in nine cases. However, in the cases where the two algorithms find the same NV, the MILP algorithm manages to optimize the second objective much better.

The results of the 100 customer instances indicates some quite significant differences in performance. In the clustered instances the MILP performs better or equal to the S-PSO in terms of number of vehicles. The S-PSO performs at best equal to the MILP and this happens in instances *C101, C104, C203* and *C204*. It should also be noted that the difference in traveled distance is significant in some instances, for example

[1]The detailed results from running Solomon's instances are available at https://github.com/carlinds/spso-vrptw.

C104 where S-PSO gives almost twice as high traveled distance as MILP for the same number of vehicles, despite the MILP reaching the solver time limit. In the random instances the S-PSO gives the best number of vehicles in 13 instances and MILP only in four instances. In instances with the same number of vehicles the MILP generally gives a lower traveled distance than the S-PSO. In the instances with mixed random clustered data there seems to be a difference between long and short scheduling horizon. In the short scheduling horizons the MILP outperforms S-PSO on number of vehicles in five out of eight instances. In the long scheduling horizons the S-PSO performs better and has a lower number of vehicles than MILP in six out of eight instances.

A comparison of the objective values obtained by each algorithm over the runtime for a few instances is given in Fig. 1 where the ratio of the objective values is plotted. The ratio is given by the objective value of MILP divided by the objective value of S-PSO which means that a ratio larger than one indicates that S-PSO found a better solution compared to MILP and vice versa. How much the ratio differs from one also indicates how much better one of the algorithms is performing compared to the other. The objective values used during the optimization are given by Equations (31) and (10) for S-PSO and MILP respectively. However to make the comparison valid the objective value used for S-PSO to calculate the ratio in Fig. 1 was the same as for MILP i.e. Equation (10).

VI. Conclusion

The results from this paper shows that both MILP and S-PSO are competitive algorithms in solving the VRPTW. The experimental results obtained from Solomon's instances show that the two algorithms perform very similar for a lower number of customers, with MILP consistently performing better in terms of minimizing the second objective, distance traveled. For the cases with higher number of customers however, i.e the 50 and 100 customer instances, the algorithms different characteristics starts to differentiate them. MILP consistently performs as good as or better than the S-PSO for the clustered data, both with short and long scheduling horizons, while the S-PSO outperforms MILP in most cases with random and mixed random clustered data with long scheduling horizons.

When comparing the two approaches, one has to bear in mind that MILP has a termination criterion based on time (600 seconds) while S-PSO's termination criterion is based on the number of iterations without improvement, therefore, for some instances, it may take much longer than 600 seconds to terminate. Also, the S-PSO can not guarantee optimality, while MILP can. If the algorithm is forcefully terminated, it will return the best solution found. In general, from the comparison of the runtimes of MILP and S-PSO, the latter shows a more consistent runtime while the MILP varies between being very quick and getting stuck at a local optima. Hence, the necessity of setting up a time limit to avoid exceedingly long run times. However, the results show that even if the MILP is interrupted, it still produces a competitive result. The results of the S-PSO could potentially be improved by increasing the stopping gap, and thus letting it run for a longer time, while an increased time limit for MILP most likely would not improve the results.

The experimental results show that the two algorithms are both competitive alternatives but perform very different depending on the problem setting. It would therefore be a good approach to select one approach depending on the problem at hand, where MILP would be favourable for clustered data and S-PSO would be a better fit for random and mixed random clustered data with longer scheduling horizons.

References

[1] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[2] Paolo Toth and Daniele Vigo. *Vehicle routing, problems, methods and applications, 2nd edition.* SIAM, 2014.

[3] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016.

[4] S Parragh, K Doerner, and R Hartl. A survey on pickup and delivery problems. part ii: Transportations between customers and depot. *Journal fur Betriebswirtschaft*, 58:82–117, 2008.

[5] Maria Gabriela S Furtado, Pedro Munari, and Reinaldo Morabito. Pickup and delivery problem with time windows: a new compact two-index formulation. *Operations Research Letters*, 45(4):334–341, 2017.

[6] Maha Gmira, Michel Gendreau, Andrea Lodi, and Jean-Yves Potvin. Tabu search for the time-dependent vehicle routing problem with time windows on a road network. *European Journal of Operational Research*, 2020.

[7] Yongbo Li, Hamed Soleimani, and Mostafa Zohal. An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives. *Journal of Cleaner Production*, 227:1161–1172, 2019.

[8] Florian Arnold and Kenneth Sörensen. Knowledge-guided local search for the vehicle routing problem. *Computers & Operations Research*, 105:32–46, 2019.

[9] Mahdi Abbasi, Milad Rafiee, Mohammad R Khosravi, Alireza Jolfaei, Varun G Menon, and Javad Mokhtari Koushyar. An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems. *Journal of Cloud Computing*, 9(1):6, 2020.

[10] Yue-Jiao Gong, Jun Zhang, Ou Liu, Rui-Zhang Huang, Henry Shu-Hung Chung, and Yu-Hui Shi. Optimizing the vehicle routing problem with time windows: a discrete particle swarm optimization approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(2):254–267, 2011.

[11] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[12] Riccardo Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008.

[13] Yudong Zhang, Shuihua Wang, and Genlin Ji. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015, 2015.

[14] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[15] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[16] Jing J Liang, A Kai Qin, Ponnuthurai N Suganthan, and S Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE transactions on evolutionary computation*, 10(3):281–295, 2006.